```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#define OUT 1

#define IN 0

#define LOW 14

#define POUT 14

#define PIN 15

/*static int GPIOExport(int pin) {

	#define BUFFER_MAX 3

	char buffer[BUFFER_MAX];

	ssize_t bytes_written;

	int fd;

	fd = open("/sys/class/gpio/export", O_WRONLY);

	if (-1 == fd) {

		fprintf(stderr, "Failed to open export for");

		return(-1);

	}

	bytes_written = snprintf(buffer, BUFFER_MAX, "%d", pin);

	write(fd, buffer, bytes_written);

	close(fd);

	return(0);

}

*/

static int GPIOExport(int pin)
```

```
{

#define BUFFER_MAX 3
  char buffer[BUFFER_MAX];
      ssize_t bytes_written;
  int fd;

fd = open("/sys/class/gpio/export", O_WRONLY); if (-1 == fd) { fprintf(stderr, "Failed to open export for writing!₩n"); return(-1);

}

bytes_written = snprintf(buffer, BUFFER_MAX, "%d", pin); write(fd, buffer, bytes_written); close(fd); return(0);

}

static int GPIODirection(int pin, int dir)

{
  static const char s_directions_str[]  = "in₩0out";

#define DIRECTION_MAX 35
char path[DIRECTION_MAX];
  int fd;
```

```
snprintf(path, DIRECTION_MAX, "/sys/class/gpio/gpio%d/direction", pin);
  fd = open(path, O_WRONLY);
if (-1 == fd) {
  fprintf(stderr, "Failed to open gpio direction for writing!₩n");
  return(-1);

}

if (-1 == write(fd, &s_directions_str[IN == dir ? 0 : 3], IN == dir ? 2 : 3)) {

fprintf(stderr, "Failed to set direction!₩n");
  return(-1);

} close(fd);
  return(0);

}

static int GPIORead(int pin) {
        #define VALUE_MAX 30
        char path[VALUE_MAX];
        char value_str[3];
        int fd;
        snprintf(path, VALUE_MAX, "/sys/class/gpio/gpio%d/value", pin);
        fd = open(path, O_RDONLY);
```

```
    if (-1 == fd) {
        fprintf(stderr, "Failed to open gpio value for reading!₩n");
        return(-1);
    }
    if (-1 == read(fd, value_str, 3)) {
        fprintf(stderr, "Failed to read value!₩n");
        return(-1);
    }
    close(fd);
    return(atoi(value_str));
}


static int GPIOWrite(int pin, int value)

{ static const char s_values_str[] = "01";

char path[VALUE_MAX];
  int fd;

snprintf(path, VALUE_MAX, "/sys/class/gpio/gpio%d/value", pin);
fd = open(path, O_WRONLY);
if (-1 == fd) {
fprintf(stderr, "Failed to open gpio value for writing!₩n");
return(-1);
```

```
}

if (1 != write(fd, &s_values_str[LOW == value ? 0 : 1], 1)) {

fprintf(stderr, "Failed to write value!₩n");
  return(-1);

}

close(fd);
  return(0);

}


static int GPIOUnexport(int pin) {
	char buffer[BUFFER_MAX];
	ssize_t bytes_written;
	int fd;
	fd = open("/sys/class/gpio/unexport", O_WRONLY);
	if (-1 == fd) {
		fprintf(stderr, "Failed to open unexport for writing!₩n");
		return(-1);
	}
	bytes_written = snprintf(buffer, BUFFER_MAX, "%d", pin);
	write(fd, buffer, bytes_written);
```

```
    close(fd);

    return(0);

}


//메인함수

int main(int argc, char *argv[])


{ int value, i;


if (-1 == GPIOExport(POUT) || -1 == GPIOExport(PIN)) return(1);


if (-1 == GPIODirection(POUT, OUT) || -1 == GPIODirection(PIN, IN)) return(2);


for(i=0; i<5; i++)


{

value = rend()% 2; //랜덤  레벨


printf("Writing %d in GPIO %d₩n", value, POUT);


if (-1 == GPIOWrite(POUT, value)) return(3);


value = GPIORead(PIN);
```

```
printf("Reading %d in GPIO %d₩n", value, PIN);

usleep(500 * 1000);

}

if (-1 == GPIOUnexport(POUT) || -1 == GPIOUnexport(PIN)) return(4);

return(0);

}
```